MASSACHUSETTS INSTITUTE OF TECHNOLOGY

VLSI PUBLICATIONS

VLSI Memo No. 88-469
August 1988

OPTIMAL SIMULATIONS BY BUTTERFLY NETWORKS

Sandeep N. Bhatt, Fan R. K. Chung, Jia-Wei Hong, F. Thomson Leighton, and Arnold L. Rosenberg

Abstract

The power of Butterfly-type networks relative to other proposed multicomputer interconnection networks is studied, by considering how efficiently the Butterfly can simulate the other networks. Simulation is represented formally via graph embeddings, so the topic here becomes: How efficiently can one embed the graph underlying a given network in the graph underlying the Butterfly network? The efficiency of an embedding of a graph G in a graph H is measured in terms of: the *dilation*, or, the maximum amount that any edge of G is "stretched" by the embedding; the *expansion*, or, the ratio of the number of vertices of H to the number of vertices of G. Three general results about embeddings in Butterfly-type graphs are established here, that expose a number of simulations by Butterfly-type networks, which are optimal (to within constant factors): (1) Any complete binary tree can be embedded in a Butterfly graph, with simultaneous dilation $O(1)$ and expansion $O(1)$. (2) Any n-vertex graph having a $\sqrt{2}$-bifurcator of size $S = \Omega(\log n)$ can be embedded in a Butterfly graph with simultaneous dilation $O(\log S)$ and expansion $O(1)$. (3) Any embedding of a planar graph G in a Butterfly graph must have dilation $\Omega\{[\log \Sigma(G)]/\Phi(G)\}$: $\Sigma(G)$ is the size of the smallest 1/3-2/3 vertex-separator of G; $\Phi(G)$ is the size of G's largest interior face. Corollaries include: (a) The n-vertex X-tree can be embedded in the Butterfly with simultaneous dilation $O(\log \log n)$ and expansion $O(1)$; no embedding yields smaller dilation, independent of expansion. (b) Every embedding of the $n \times n$ mesh in the Butterfly has dilation $\Omega(\log n)$; any expansion-$O(1)$ embedding of the mesh in the Butterfly achieves this dilation. These results, which extend to Butterfly-like graphs such as the Cube-Connected Cycles and Benes networks, supply the first examples of graphs that can be embedded more efficiently in the Hypercube than in the Butterfly.

Keywords: computer networks, interconnection networks, parallel computer systems, Butterfly networks, graph embeddings

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

Acknowledgements

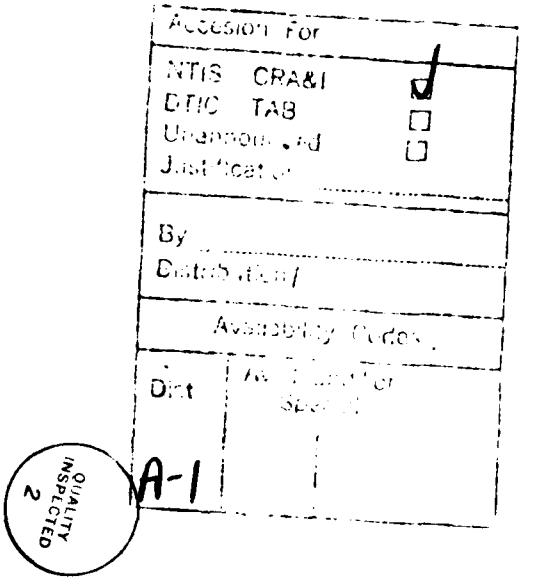
Presented at the 20th ACM Symposium on Theory of Computing, Chicago, IL, May 2-4, 1988. This work was supported in part by NSF Grant Nos. MIP-86-01885, DCI-85-04308, and DCI-87-96236, Air Force Contract OSR-86-0076, the Defense Advanced Research Projects Agency under contract nos. N00014-80-C-0622 and N00014-87-K-0825, an NSF Presidential Young Investigators Award, with matching funds from IBM and AT&T.

Author Information

Bhatt: Department of Computer Science, Yale University, New Haven, CT 06520; Chung: Mathematics, Information Sciences and Operations Research Division, Bell Communications Research, Morristown, NJ 07960; Hong: Beijing Computer Institute, Beijing 10044, CHINA; Leighton: Department of Mathematics, MIT, Cambridge, MA 02139; Rosenberg: Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003.

OPTIMAL SIMULATIONS BY BUTTERFLY NETWORKS[1]

Sandeep N. Bhatt†, Fan R.K. Chung§,
Jia-Wei Hong‡, F. Thomson Leighton¶,
Arnold L. Rosenberg
Computer and Information Science Department
University of Massachusetts



†Yale University, New Haven CT
§Bell Communications Research, Morristown, NJ
‡Beijing Computer Institute, Beijing, CHINA
¶MIT, Cambridge, MA

[1] A preliminary version of this paper was presented at the *20th ACM Symposium on Theory of Computing*, Chicago, IL, May 2-4, 1988

Contact Author:
Arnold L. Rosenberg
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Acknowledgments of Support:
The research of S. N. Bhatt was supported in part by NSF Grant MIP-86-01885; the research of F.T. Leighton was supported in part by Air Force Contract OSR-86-0076, DARPA Contract N00014-80-C-0622, Army Contract DAAL-03-86-K-0171, and and NSF Presidential Young Investigator Award with matching funds from ATT and IBM; the research of A. L. Rosenberg was supported in part by NSF Grants DCI-85-04308 and DCI-87-96236.

Authors' Present Addresses:
Sandeep N. Bhatt: Department of Computer Science, Yale University, New Haven, CT 06520;
Fan R. K. Chung: Mathematics, Information Sciences and Operations Research Division, Bell Communications Research, Morristown, NJ 07960;
Jia-Wei Hong: Beijing Computer Institute, Beijing 10044, CHINA; currently visiting at Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003;
F. Thomson Leighton: Department of Mathematics, MIT, Cambridge, MA 02139;
Arnold L. Rosenberg: Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003

Categories and Subject Descriptors: C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures – *interconnection architectures*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems – *Computations on discrete structures*; G.2.1 [**Discrete Mathematics**]: Combinatorics – *combinatorial algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory – *graph algorithms*

General Terms: Algorithms, Design, Theory

Additional Key Words and Phrases: Mapping algorithms, mapping problems, parallel architectures, processor arrays, simulation

Abstract

The power of Butterfly-type networks relative to other proposed multicomputer interconnection networks is studied, by considering how efficiently the Butterfly can simulate the other networks. Simulation is represented formally via graph embeddings, so the topic here becomes: How efficiently can one embed the graph underlying a given network in the graph underlying the Butterfly network? The efficiency of an embedding of a graph G in a graph H is measured in terms of: the *dilation*, or, the maximum amount that any edge of G is "stretched" by the embedding; the *expansion*, or, the ratio of the number of vertices of H to the number of vertices of G. Three general results about embeddings in Butterfly-type graphs are established here, that expose a number of simulations by Butterfly-type networks, which are optimal (to within constant factors): (1) Any complete binary tree can be embedded in a Butterfly graph, with simultaneous dilation $O(1)$ and expansion $O(1)$. (2) Any n-vertex graph having a $\sqrt{2}$-bifurcator of size $S = \Omega(\log n)$ can be embedded in a Butterfly graph with simultaneous dilation $O(\log S)$ and expansion $O(1)$. (3) Any embedding of a planar graph G in a Butterfly graph must have dilation $\Omega\left(\frac{\log \Sigma(G)}{\Phi(G)}\right)$: $\Sigma(G)$ is the size of the smallest 1/3-2/3 vertex-separator of G; $\Phi(G)$ is the size of G's largest interior face. Corollaries include: (a) The n-vertex X-tree can be embedded in the Butterfly with simultaneous dilation $O(\log \log n)$ and expansion $O(1)$; no embedding yields smaller dilation, independent of expansion. (b) Every embedding of the $n \times n$ mesh in the Butterfly has dilation $\Omega(\log n)$; any expansion-$O(1)$ embedding of the mesh in the Butterfly achieves this dilation. These results, which extend to Butterfly-like graphs such as the Cube-Connected Cycles and Benes networks, supply the first examples of graphs that can be embedded more efficiently in the Hypercube than in the Butterfly.

1. INTRODUCTION

This paper reports on a continuing program of the authors, dedicated to determining the relative computational capabilities of the various interconnection networks that have been proposed for use as multicomputer interconnection networks [BCLR, BI, GHR, Le]. We focus here on one member of the family of *butterfly*-like machines, that have become one of the benchmark architectures for multicomputers. The major contributions of this paper are the following general results about embeddings of graphs in Butterfly networks[1]:

1. We embed the complete binary tree in the Butterfly network, with simultaneous dilation $O(1)$ and expansion $O(1)$.
2. We embed any n-vertex graph having a $\sqrt{2}$-bifurcator of size $S = \Omega(\log n)$ in the Butterfly network, with simultaneous dilation $O(\log S)$ and expansion $O(1)$.
3. We prove that any embedding of any planar graph G in a Butterfly network must have dilation
$$\Omega\left(\frac{\log \Sigma(G)}{\Phi(G)}\right)$$
where: $\Sigma(G)$ is the size of the smallest 1/3-2/3 vertex-separator of G; $\Phi(G)$ is the size of G's largest interior face.

The latter two results lead to embeddings of graphs such as X-trees and meshes in the Butterfly, that are optimal, to within constant factors. By Result 2, such embeddings can be found with expansion $O(1)$ and with, respectively, dilation $O(\log \log n)$ and $O(\log n)$; by Result 3, no embeddings can improve on these dilations, independent of expansion. These embeddings expose X-trees and meshes as the *first known graphs that can be embedded very efficiently in the Hypercube* (simultaneous dilation $O(1)$ and expansion $O(1)$) *but have no efficient embedding in butterfly-like graphs.* Note that, if we restrict attention only to the issue of dilation, then – to within constant factors – these graphs cannot be embedded any more efficiently in Butterfly graphs than they can in complete binary trees!

1.1. The Formal Setting

The technical vehicle for our investigations is the following notion of graph embedding [Ro]. Let G and H be simple undirected graphs. An *embedding* of G in H is a

[1] All technical terms are defined in Section 1.1.

one-to-one association of the vertices of G with vertices of H, plus a *routing* of each edge of G within H, i.e., an assignment of a path in H connecting the images of the endpoints of each edge of G. The *dilation* of the embedding is the length of the longest path in H that routes an edge of G; it thus measures how much the edges of G are "stretched" by the embedding. The *expansion* of the embedding is the ratio $|H|/|G|$ of the number of vertices in $H$to the number of vertices in G. We use the dilation- and expansion-costs of the best embedding of G in H as our measures of how well H can *simulate* G as an interconnection network: One views the graph H as abstracting the processor-intercommunication structure of a physical architecture; one views the graph G as abstracting either the task-interdependency structure of an algorithm one wants to implement on H or the processor-intercommunication structure of an architecture one wants to simulate on H.

Remark. A third important measure of how well H can simulate G is *congestion*, the maximum number of edges that are routed through a single edge (or vertex) of H. Congestion does not play a major role in this paper, however, since
1. our embedding of a complete binary tree in a Butterfly trivially has unit congestion;
2. the n-vertex Butterfly is known to be able to simulate any n-vertex bounded-degree graph with $O(\log n)$ delay, irrespective of the fact that the dilation and congestion of the corresponding embedding may both be $\Omega(\log n)$; 3. our major focus is on developing broadly applicable techniques for bounding the dilation of embeddings.
Hence, for our purposes, dilation is the central measure of concern.

Our results hold for a large variety of "levelled" Hypercube-derivative host graphs (which play the role of our H's), that we collectively term *butterfly* networks. For the sake of rigor, we focus on one particular such network (which can be viewed as the FFT network, with input and output vertices identified), although we could just as easily substitute other such graphs – the Cube-Connected Cycles [PV] or Benes network [Be], for example. Formally,

- Let m be a positive integer. The *m-level Butterfly graph* $B(m)$ has vertex-set[2]

$$V_m = \{0, 1, \cdots, m-1\} \times \{0,1\}^m.$$

  The subset $V_{m,\ell} = \{\ell\} \times \{0,1\}^m$ of V_m $(0 \leq \ell < m)$ is the ℓ^{th} *level* of $B(m)$. The string $x \in \{0,1\}^m$ of vertex $\langle \ell, x \rangle$ is the *position-within-level string* (*PWL string*, for short) of the vertex. The edges of $B(m)$ form *butterflies* (or, copies

[2] $\{0,1\}^m$ denotes the set of length-m binary strings.

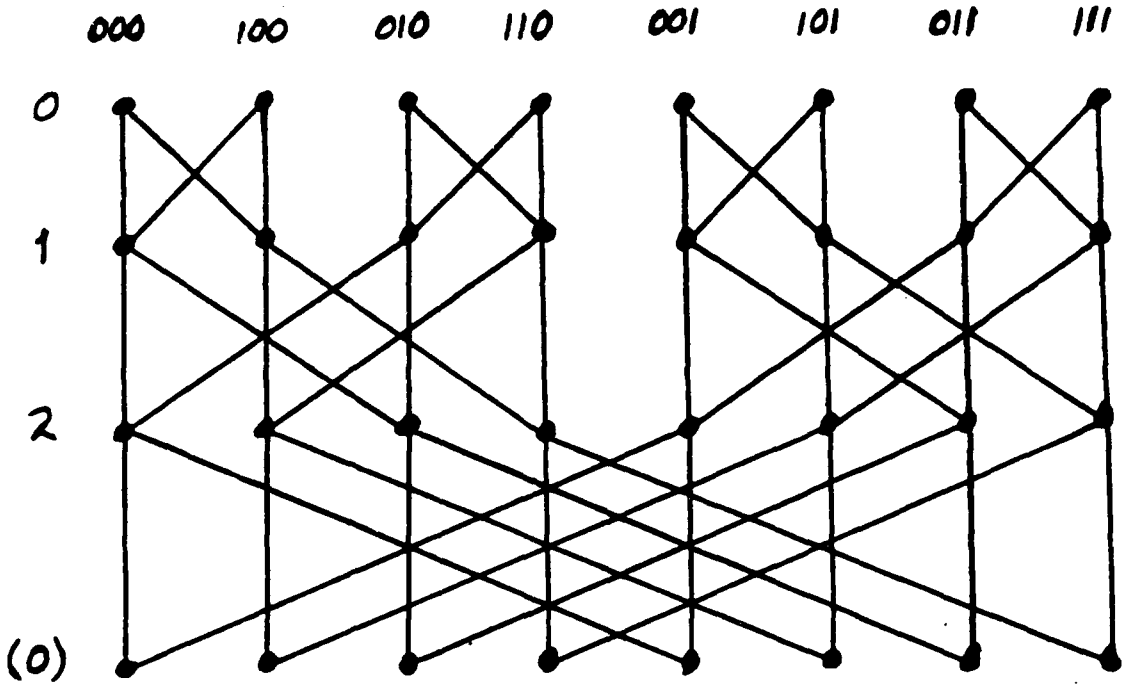


Figure 1: The 3-level Butterfly graph $B(3)$

of $K_{2,2}$) between consecutive levels of vertices, with wraparound in the sense that level 0 is identified with level m. Each butterfly connects vertices

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}0\beta_{\ell+1}\cdots\beta_{m-1}\rangle$$

and

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}1\beta_{\ell+1}\cdots\beta_{m-1}\rangle$$

on level ℓ of $B(m)$ ($0 \le \ell < m$; each $\beta_i \in \{0,1\}$) with vertices

$$\langle \ell+1 (\mathrm{mod}\ m),\ \beta_0\beta_1\cdots\beta_{\ell-1}0\beta_{\ell+1}\cdots\beta_{m-1}\rangle$$

and

$$\langle \ell+1 (\mathrm{mod}\ m),\ \beta_0\beta_1\cdots\beta_{\ell-1}1\beta_{\ell+1}\cdots\beta_{m-1}\rangle$$

on level $\ell+1 (\mathrm{mod}\ m)$ of $B(m)$. One can represent $B(m)$ level by level, in such a way that at each level the PWL strings are the reversals of the binary representations of the integers $0, 1, \ldots, 2^m - 1$, in that order. See Fig. 1.

The guest graphs in our study, which play the role of our G's, are complete binary trees, X-trees, and meshes; see Fig. 2. Formally,

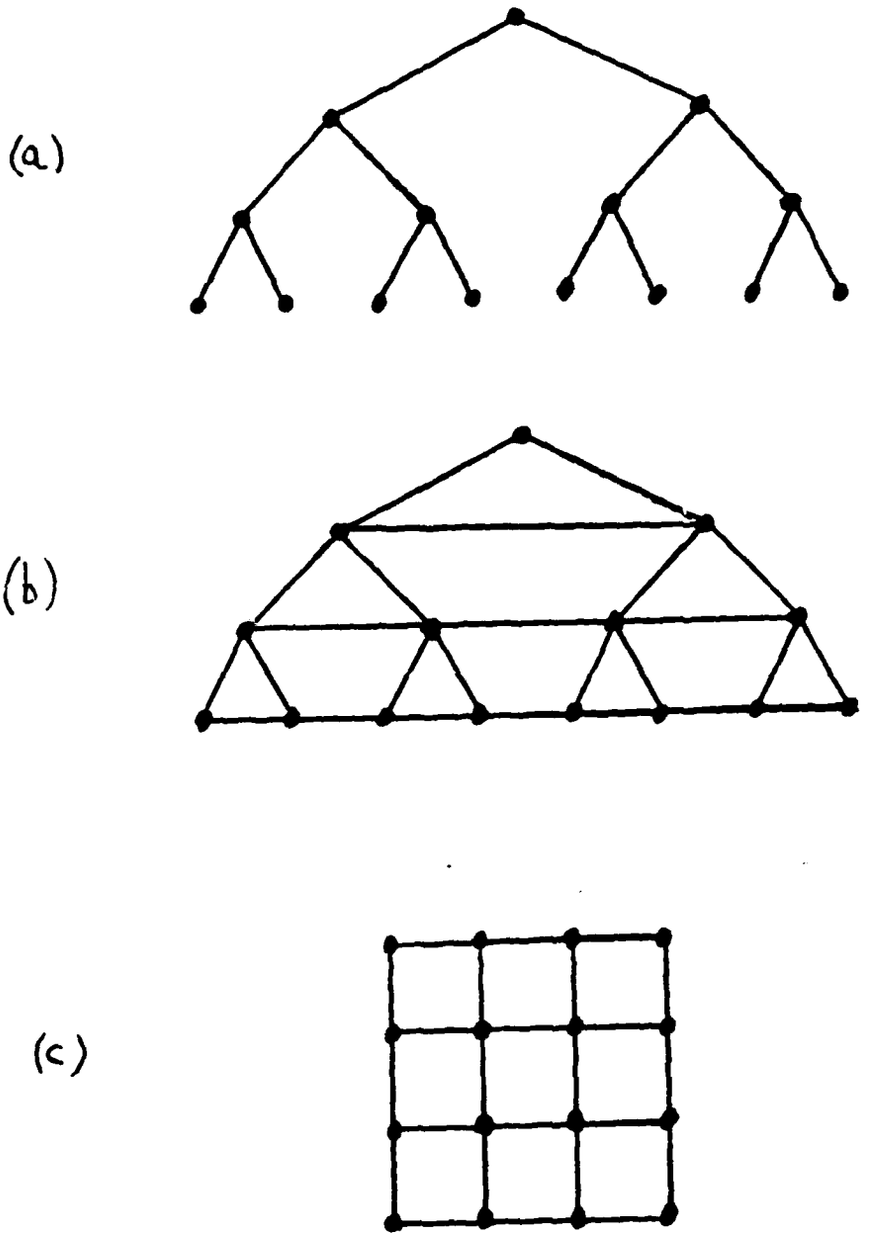


Figure 2: The Complete Binary tree $T(2)$, the X-tree $X(2)$, and the mesh $M(4)$

- The *height-h complete binary tree* $T(h)$ is the graph whose $(2^{h+1}-1)$-element vertex-set comprises all binary strings of length at most h, and whose edges connect each vertex x of length less than h with vertices $x0$ and $x1$. The (unique) string of length 0 is the *root* of the tree, which is the sole occupant of level 0 of the tree; the 2^ℓ strings of length ℓ are the *level-ℓ* vertices of the tree; the strings of length h (i.e., the level-h vertices) are the *leaves* of the tree.

- The *height-h X-tree* $X(h)$ is the graph that is obtained from the height-h complete binary tree $T(h)$ by adding *cross edges* connecting the vertices at each level of $T(h)$ in a path, with the vertices in lexicographic order. X-trees inherit a level structure from their underlying complete binary trees.

- The *$s \times s$ mesh* $M(s)$ is the graph whose s^2-element vertex set comprises the ordered pairs of integers

$$\{1, 2, \cdots, s\} \times \{1, 2, \cdots, s\},$$

and whose edges connect vertices $\langle a, b\rangle$ and $\langle c, d\rangle$ just when $|a-c|+|b-d| = 1$.

All of these networks have been seriously proposed as interconnection networks for multicomputers [DP, Ga, HZ], hence are important candidates for our study. Another approach to comparing these networks, via implementation and analysis of specific algorithms, appears in [Ag].

Our results depend on three structural features of a graph G:

1. Let S and k be positive integers. The n-vertex graph G has a *k-color $\sqrt{2}$-bifurcator of size S* if either $n < 2$ or the following holds for every way of labelling each vertex of G with one of k possible labels: By removing $\leq S$ vertices from G, one can partition G into subgraphs G_1 and G_2 such that[3]
   (a) $\big|\,|G_1| - |G_2|\,\big| \leq 1$.
   (b) For each label l, the number of l-labelled vertices in G_1 is within 1 of the number of l-labelled vertices in G_2.
   (c) Each of G_1 and G_2 has a k-color $\sqrt{2}$-bifurcator of size $S/\sqrt{2}$.

2. A *1/3-2/3 (vertex-)separator* of G is a set of vertices whose removal partitions G into subgraphs, each having $\geq |G|/3$ vertices; we denote by $\Sigma(G)$ the size of the smallest 1/3-2/3 vertex-separator of G.

3. When G is planar and we are given a witnessing planar embedding ϵ, we denote by $\Phi_\epsilon(G)$ the number of vertices in G's largest interior face in the embedding. When ϵ is clear from context, we omit the subscript.

[3] We denote by $|G|$ the number of vertices in the graph G.

1.2. The Main Results

We prove three results about optimal embeddings in the Butterfly that lead to a variety of nontrivial optimal embeddings.

Theorem 1 *The complete binary tree $T(h)$ can be embedded in a Butterfly graph, with simultaneous dilation $O(1)$ and expansion $O(1)$.*

Obviously, the embedding of Theorem 1 is within a constant factor of optimal in both dilation and expansion. Building on the embedding, we obtain the following general upper bound result.

Theorem 2 *Any n-vertex graph G having a $\sqrt{2}$-bifurcator of size $S = \Omega(\log n)$ can be embedded in a Butterfly graph with simultaneous dilation $O(\log S)$ and expansion $O(1)$.*

We balance Theorem 2 with one of the first broadly applicable results for bounding dilation from below.

Theorem 3 *Any embedding of a nontree planar graph G in a Butterfly graph has dilation $\Omega\left(\frac{\log \Sigma(G)}{\Phi(G)}\right)$. This bound cannot be improved in general.*

Direct application of the proofs of these results yields the following optimal embeddings.

Corollary 1 *The height-h X-tree $X(h)$ can be embedded in a Butterfly graph with simultaneous dilation $O(\log h) = O(\log \log |X(h)|)$ and expansion $O(1)$. Any embedding of $X(h)$ in a Butterfly graph must have dilation $\Omega(\log h) = \Omega(\log \log |X(h)|)$.*

Corollary 2 *Any embedding of the $s \times s$ mesh $M(s)$ in a Butterfly graph must have dilation $\Omega(\log s) = \Omega(\log |M(s)|)$.*

Corollary 2 betokens a mismatch in the structures of meshes and Butterfly graphs, since *any* expansion-$O(1)$ embedding of any graph G in $B(m)$ has dilation $O(\log |G|)$.[4]

[4]This follows from the facts that $B(m)$ has $m2^m$ vertices and diameter $O(m)$.

Theorem 1 and Corollaries 1 and 2 can be interpreted as yielding tight bounds on the efficiency with which a Butterfly machine can simulate a complete-binary-tree machine, an X-tree machine, and a mesh-structured machine, with regard to both delay (dilation) and resource utilization (expansion). Equating dilation with delay is most appropriate when the machines are to be run in SIMD mode.

The next three sections are devoted to proving our main results.

2. COMPLETE BINARY TREES

2.1. Embedding Many Small Trees in a Butterfly

It is obvious from inspection that one can find an instance of the height-$(m-1)$ complete binary tree $T(m-1)$ rooted at every vertex of $B(m)$. Somewhat less obvious is the fact that one can find m mutually disjoint instances of $T(m-1)$ as subgraphs of $B(m)$. We now verify this fact via an embedding which will prove useful as we develop our final embedding.

Proposition 1 *For every integer m, one can find m mutually disjoint instances of $T(m-1)$ as subgraphs of $B(m)$.*

Proof. To simplify exposition, we represent sets of binary strings by strings over the alphabet $\{0,1,*\}$, using $*$ as a wild-card character. The length-k string

$$\beta = \beta_0\beta_1\cdots\beta_{k-1},$$

where each $\beta_i \in \{0,1,*\}$, represents the set $\sigma(\beta)$ of all length-k binary strings that have a 0 in each position i of β where $\beta_i = 0$, a 1 in each position i of β where $\beta_i = 1$, and either a 0 or a 1 in each position i of β where $\beta_i = *$. For illustration, $\sigma(010) = \{010\}$, and $\sigma(0*1) = \{001,011\}$. Call the string β the *code* for the set $\sigma(\beta)$.

On to our embeddings of m instances of $T(m-1)$ in $B(m)$: For any letter a and nonnegative integer k, we denote by a^k a string of k a's.

For the first instance of $T(m-1)$, we have the following correspondence between tree vertices and Butterfly vertices.

$T(m-1)$	$B(m)$
level 0	$\langle 0,\ 0^m \rangle$
level 1	$\langle 1,\ *0^{m-1} \rangle$
level 2	$\langle 2,\ *^2 0^{m-2} \rangle$
$\vdots$	$\vdots$
level $m-1$	$\langle m-1,\ *^{m-1}0 \rangle$

For each subsequent instance of $T(m-1)$, say the j^{th} where $1 < j < m$, we have the following correspondence between tree vertices and Butterfly vertices.

$T(m-1)$	$B(m)$
level 0	$\langle j-1,\ 0^{j-1}10^{m-j-1}1 \rangle$
level 1	$\langle j,\ 0^{j-1}1*0^{m-j-2}1 \rangle$
level 2	$\langle j+1 (\text{mod } m),\ 0^{j-1}1*^2 0^{m-j-3}1 \rangle$
$\vdots$	$\vdots$
level $m-1$	$\langle j-2,\ *^{j-1}1*^{m-j} \rangle$

The placement of the 1's in the PWL strings ensures that the m instances of $T(m-1)$ are mutually disjoint. To verify this, via contradiction, let us look at an arbitrary level ℓ of $B(m)$ and at arbitrary distinct tree vertices i and j that collide at some position within level ℓ of $B(m)$. It is clear that all Butterfly vertices that are images of the same instance of $T(m-1)$ are distinct, so we may assume that vertices i and j come from distinct instances of $T(m-1)$, call them $\iota(i)$ and $\iota(j)$, where the ι-"name" of an instance of $T(m-1)$ is the level of $B(m)$ where its root resides. We consider four cases that exhaust the possibilities. In each case, we adduce a property of the PWL strings that precludes any overlap in the images of the trees.

$\iota(i) = 0$:

> If $\iota(i) = 0$, then the PWL string of i ends with $0^{m-\ell}$, while the PWL string of j has a 1 in this range, specifically, in position $m-1$ if $j \leq \ell$, and in position j if $j > \ell$.

$1 \leq \iota(i) < \iota(j) \leq \ell$:

Every PWL string of $\iota(i)$ starts with 0^i1, while every PWL string of $\iota(j)$ starts with 0^j1.

$1 \leq \iota(i) \leq \ell < \iota(j)$:

Every PWL string of $\iota(i)$ has a 0 in position j, while every PWL string of $\iota(j)$ has a 1 in that position.

$\ell < \iota(i) < \iota(j) < m$:

Every PWL string of $\iota(i)$ has a 1 in position i, while every PWL string of $\iota(j)$ has a 0 in position i.

The proof is complete. □

An algebraic proof of Proposition 1, which is "cleaner" than our combinatorial proof here, appears in [ABR]; however, it is the embedding rather than the result that will be helpful in our proof of Theorem 1.

The embedding in our proof of Proposition 1 does not serve us directly in our attempt to embed a large complete binary tree in a small Butterfly, since (for one thing) it places the roots of every instance of $T(m-1)$ at a different level of $B(m)$; and it is not clear how to combine these instances into a bigger complete binary tree with small dilation. However, the overall strategy of the embedding will be useful in Section 2.2.*D*.

2.2. Optimally Embedding Trees in Butterfly Graphs

We turn now to the proof of Theorem 1. Specifically, we prove the following.

For any integer m, one can embed the complete binary tree $T(m + \lfloor \log m \rfloor - 1)$ in the Butterfly graph $B(m+3)$, with dilation $O(1)$.

To simplify our description, let $q =_{\text{def}} m + \lfloor \log m \rfloor - 1$, and assume henceforth that m is even; clerical changes will remove the assumption.

A. The Embedding Strategy

We wish to embed the tree $T(q)$ with dilation $O(1)$, in the smallest Butterfly that is big enough to hold the tree, namely, $B(m)$. We fall somewhat short of this

goal, but not by much: We find an embedding with dilation $O(1)$, but we have to use a somewhat larger host Butterfly graph (specifically, $B(m+3)$) in order to resolve collisions in our embedding procedure. Our embedding proceeds in four stages. Stage 1 embeds the top $\log m$ levels of $T(q)$ with unit dilation in $B(m)$, thereby specifying implicitly the images in $B(m)$ of the roots of the $m/2$ subtrees of $T(q)$ rooted at level $\log m - 1$. Stage 2 expands these subtrees a further $m/2$ levels, but now in $B(m+1)$, with dilation 2, thereby specifying implicitly the images in $B(m)$ of the roots of the $m \cdot 2^{m/2-1}$ subtrees of $T(q)$ rooted at level $m/2 + \log m - 1$ of the tree. In Stage 3, we embed the final $m/2$ levels of $T(q)$ in $B(m+1)$, with dilation 4. The vertex-mappings in each stage are embeddings (i.e., are one-to-one); there is, however, "overlap" (i.e., distinct vertices of $T(q)$ getting mapped to the same vertex of $B(m+1)$) among the mappings of the three stages. In Stage 4, we eliminate this overlap by expanding the host Butterfly by two more levels, thereby giving us four connected isomorphic copies of $B(m+1)$. At the cost of increasing dilation by 2, we modify our mapping so that each of Stages 1, 2, 3 is performed in a distinct copy of $B(m+1)$, thereby eliminating all overlap.

B. Stage 1: The Top $\log m$ Levels of $T(q)$

We place the root of $T(m + \log m)$ at position

$$\langle m - \log m,\ 0^m \rangle$$

of $B(m)$. We then proceed to higher-numbered levels, embedding the top $\log m$ levels of $T(q)$ as a subgraph of $B(m)$, ending up with the leaves of these levels in positions

$$\langle 0,\ 0^{m-\log m+1}{}_*{}^{\log m-1} \rangle$$

of $B(m)$ (because of wraparound). See Fig. 3. We call the rightmost $\log m - 1$ bits of each of the resulting PWL strings the *signature* of the Butterfly position and of the subtree rooted at that position. It is convenient to interpret a signature as an integer in the range $\{0, 1, \cdots, m/2 - 1\}$, as well as a bit string.

The embedding in Stage 1 is trivially one-to-one, with unit dilation.

C. Stage 2: The Next $m/2$ Levels of $T(q)$

Call the $(m/2+1)$-level subtree of $T(q)$ that has signature k, the k^{th} *subtree*. Our goal is to embed the k^{th} subtree in $B(m+1)$ (with dilation 2), so that its $2^{m/2}$ leaves form the set of positions[5]

$$\langle m-1,\ *0*0\cdots*0*1*0\cdots*0*0? \rangle,$$

[5]The last bit position is not affected by this Stage, so is denoted "?".

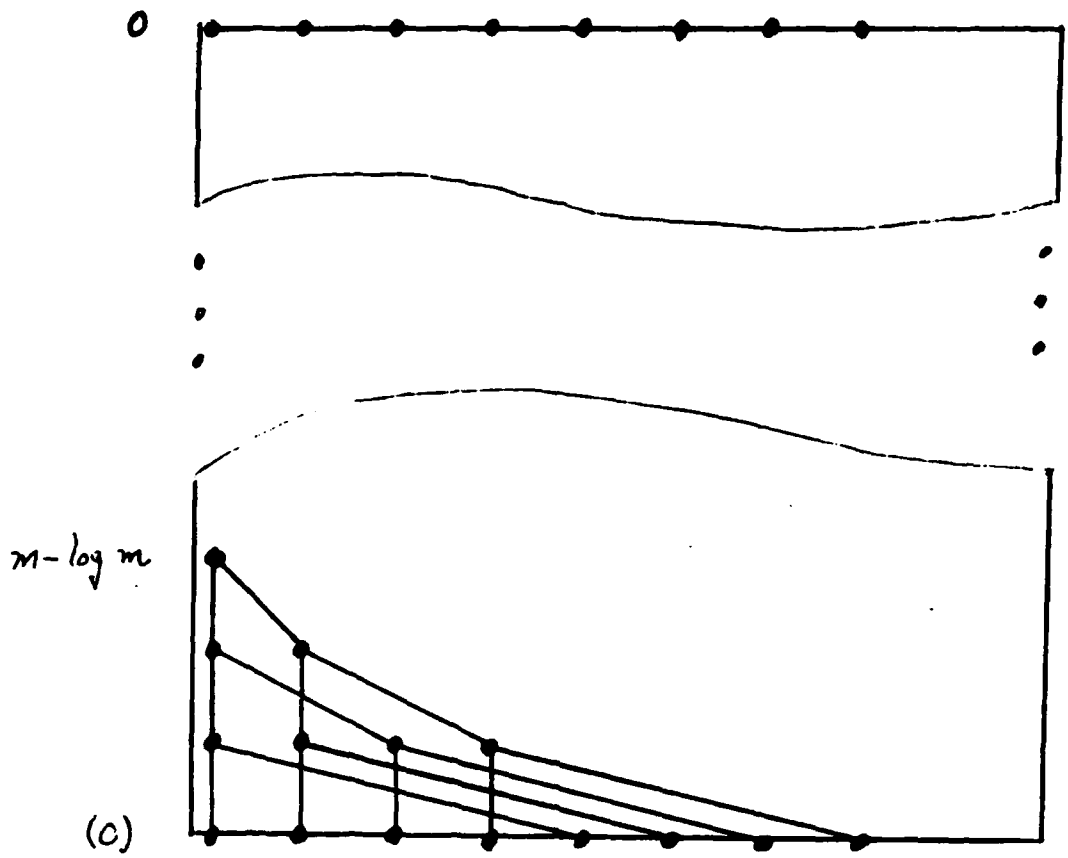


Figure 3: A logical view of the first $\log m$ levels of the embedding

where the 1 appears in the k^{th} even position from the right (using 0-based counting); call this the *signatory* 1 of the tree position. For instance, when $m = 8$, the second subtree has leaves in positions

$$\langle 7,\ *0*1*0*0?\rangle$$

of $B(9)$. We embed these $(m/2+1)$-level trees by alternating binary and unary branchings in $B(m+1)$, starting at the "roots" placed at level-0 vertices of $B(m+1)$ during Stage 1; we place a tree-vertex after each unary branching. See Fig. 4. Binary branchings generate the $*$'s in the code for the set of PWL strings, while unary branchings generate the 0's and 1's in the code. As a simple example: a binary branching from vertex

$$\langle 0,\ 000000011\rangle,$$

which holds the root of one of the subtrees planted during Stage 1, generates vertices

$$\langle 1,\ *00000011\rangle;$$

a unary branching thence generates vertices

$$\langle 2,\ *00000011\rangle,$$

where we place the level-1 vertices of the subtree; a second binary branching generates vertices

$$\langle 3,\ *0*000011\rangle;$$

a unary branching thence generates vertices

$$\langle 4,\ *0*100011\rangle,$$

where we place the level-2 vertices of the subtree; a subsequent sequence of alternating binary and unary branchings finally embeds the desired set of leaf positions in the advertised vertices of $B(m+1)$.

This stage of our embedding clearly has dilation 2. The fact that that this stage is one-to-one (though it may produce conflicts with the embedding from Stage 1) has two origins. First, we are using levels 0 through m of $B(m+1)$ for the $m+1$ levels of this stage, so the leaves of the embedded trees do not wrap around to conflict with their roots. Second, each signatory 1, whose placement identifies its respective tree, is set "on" *before* the signature bits are reached and altered by the sequence of branchings. This is ensured by the fact that we place the signatory 1 by counting from the right: the signature bits occupy the rightmost $\log m - 1$ bits

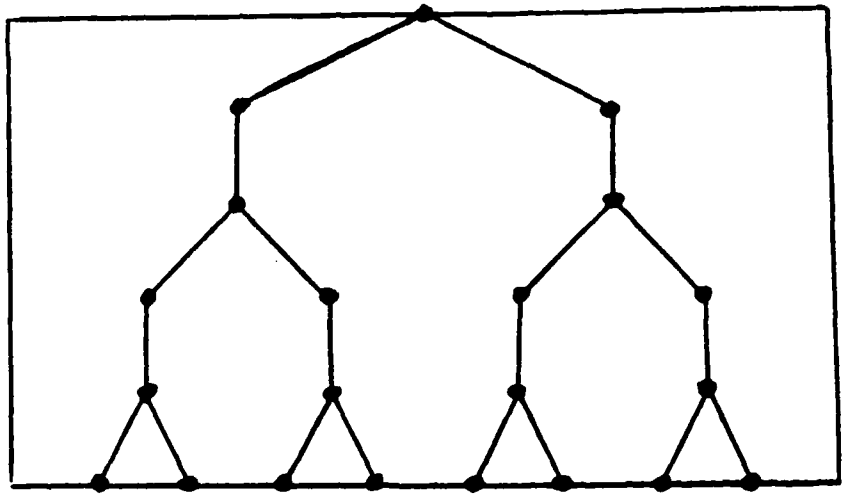


Figure 4: A logical view of the next $m/2$ levels of the embedding

of the PWL string; by the time the branchings have reached the i^{th} bit from the right, only the rightmost $(\log i)$ bits of the signature are needed to specify the next position where branching occurs. Hence, at the point when we place the signatory 1 in the i^{th} position, the odd-numbered positions to the left of the 1 are all 0, and the positions to the right of the 1 form the binary representation of i, possibly with leading 0's.

D. Stage 3: The Final $m/2$ Levels of $T(q)$

Our goal in Stage 3 is to use the $m \cdot 2^{m/2-1}$ leaves of the $m/2$ trees generated in Stage 2 as the roots of the $(m/2+1)$-level subtrees comprising the bottom $m/2$ levels of $T(q)$. Each root has a signatory 1, identifying the subtree it came from in Stage 2, and a *serial number* obtained from the odd-numbered bits of its PWL string. The signatory 1's will keep trees sired by different Stage-2 trees disjoint; the serial numbers will guard against collisions among trees that were sired by the same Stage-2 tree. The main challenge here is to achieve the embedding while the roots of all the trees reside at the same level of $B(m+1)$ (which is how Stage 2 has placed them). To accomplish this, we have the trees grow *upward*, in the direction of lower level-numbers, for varying amounts of time, before starting to grow *downward*, in the direction of higher level-numbers. While growing either upward or downward, a tree grows via alternating unary and binary branchings, *so as to preserve the serial number*; this alternation will incur dilation 2. An additional dilation of 2 is incurred while a tree grows upward: each tree begins to grows upward using only

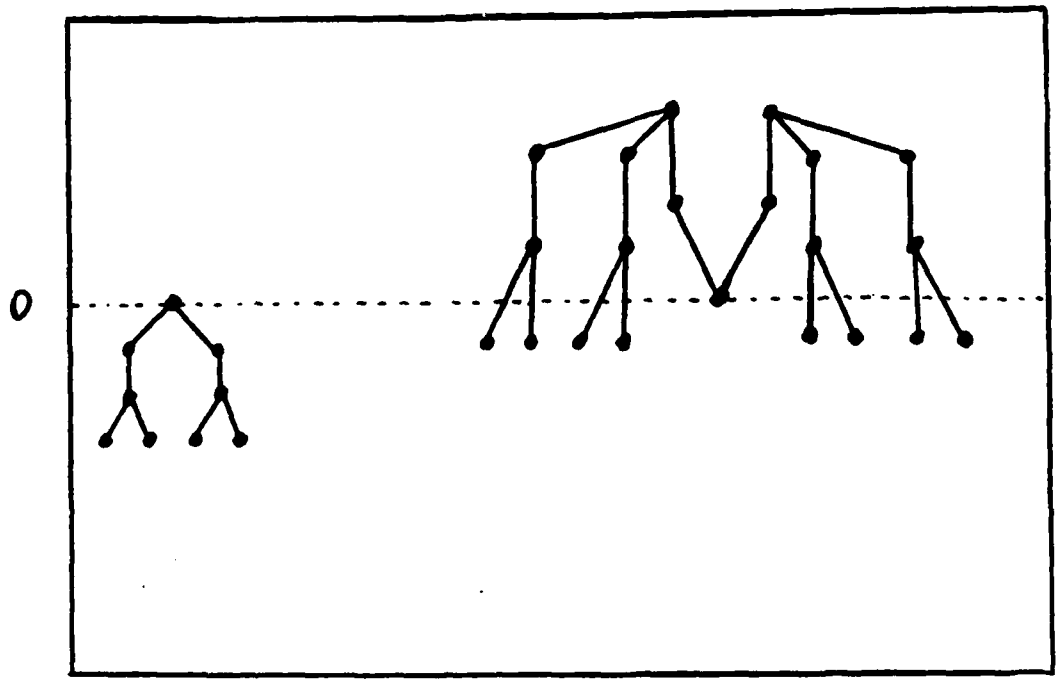


Figure 5: A logical view of the final $m/2$ levels

every fourth level of $B(m+1)$; when it "turns" from growing upward to growing downward, it uses the levels it has skipped while moving upward to regain level 0 of $B(m+1)$, at which time it grows downward using every other level of $B(m+1)$. See Fig. 5. Thus, in all, this Stage of the embedding incurs dilation 4.

All trees with the same signatory 1 (i.e., rooted at the leaves of the same Stage-2 tree) will grow in lockstep. We refer to the trees sharing a signatory 1 in the k^{th} even bit-position as the k^{th} *subtrees* of $T(q)$, $0 \leq k < m/2$. We place the vertices of the k^{th} subtrees of $T(q)$ into $B(m+1)$ as follows:

- For the 0^{th} trees, we place the 2^ℓ level-ℓ vertices of $T(q)$ at level 2ℓ of $B(m+1)$. (Thus, these trees grow downward immediately.)
- For the k^{th} trees, $k > 0$:
  - we place their unique level-0 vertex at level 0 of $B(m+1)$ (in fact this was placed during Stage 2)

- for $1 \leq \ell \leq \lfloor k/2 \rfloor$, we place their 2^ℓ level-ℓ vertices at level $m - 4\ell + 1$ of $B(m+1)$
- *if k is odd*, we place their $2^{\lceil k/2 \rceil}$ level-$(\lceil k/2 \rceil)$ vertices at level $m - 4\lceil k/2 \rceil + 3$ of $B(m+1)$
- for $\lceil k/2 \rceil + 1 \leq \ell \leq k$, we place their 2^ℓ level-ℓ vertices at level $m - 4(k - \ell) - 1$ of $B(m+1)$

Now we verify that the described mapping is one-to-one, hence an embedding. We consider separately the two potential sources of collisions.

First, we note that there can be no collisions among the $2^{m/2}$ k^{th} trees, for any k, since each of these trees has a unique serial number.

Second, we note that, for each fixed serial number, there can be no collision between the j^{th} and k^{th} trees having that serial number. This is argued most easily by considering how such trees are laid out level by level. To simplify exposition, we present only the even bit-positions of the image vertices in $B(m+1)$, since the odd bit-positions hold identical serial numbers. Note first that the top k levels of each k^{th} tree are placed in vertices of the form

$$\langle \ell,\ 0^{m/2-k}1*^k \rangle$$

in $B(m+1)$; hence, their membership in a k^{th} tree is announced by the leftmost $m/2 - k + 1$ even bit-positions of the PWL strings. For tree-levels $> k$, the j^{th} and k^{th} trees are distinguished as follows. Say, with no loss of generality, that $j < k$. For each $0 \leq \ell \leq m/2 - k$, the level-$(k+\ell)$ vertices of each k^{th} tree are placed at vertices

$$\langle \ell,\ *^\ell 0^{m/2-k-\ell}10^k \rangle$$

of $B(m+1)$. By the same token, for each $0 \leq \ell \leq m/2 - j$, the level-$(j+\ell)$ vertices of each j^{th} tree are placed at vertices

$$\langle \ell,\ *^\ell 0^{m/2-j-\ell}10^j \rangle$$

of $B(m+1)$. Since $j < k$ by hypothesis, we see that, at those levels of $B(m+1)$ where we place vertices of both trees, the k^{th} even bit-position from the right of each k^{th} tree contains a 1, while the corresponding bit-position of each j^{th} tree contains a 0.

Thus, the mapping in this stage is an embedding.

E. Resolving Collisions

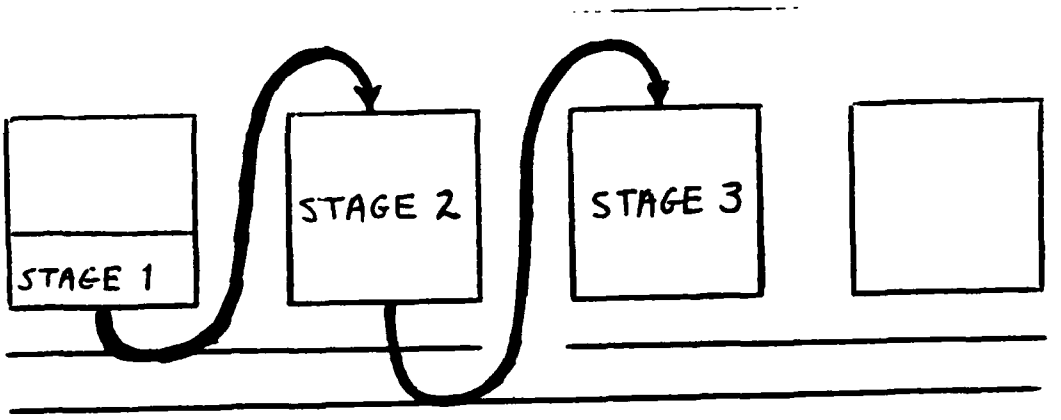


Figure 6: Replicating $B(m+1)$ to avoid collisions

We now have three subembeddings that accomplish the desired task, except for the fact that Stage i and Stage j may map different tree vertices to the same Butterfly vertex. We resolve these possible collisions as follows. Instead of performing the subembeddings in $B(m+1)$, we perform them in $B(m+3)$, placing each subembedding in a distinct copy of $B(m+1)$. We make the transition between copies of $B(m+1)$ as follows. As the Stage-1 embedding of the top of $T(q)$ reaches level $m-1$ of its copy of $B(m+1)$, we use a sequence of unary branchings in $B(m+3)$ to reach level 0 of the next copy of $B(m+1)$. We perform the Stage-2 subembedding within this second copy; this takes us to level $m-1$ of that copy, where a sequence of unary branchings in $B(m+3)$ takes us to level 0 of the third copy of $B(m+1)$. We perform the Stage-3 subembedding in this third copy. See Fig. 6. The transition from level $m-1$ of the second copy of $B(m+1)$ to level 0 of the third copy engenders dilation 4.

The embedding, hence the proof, is now complete. □

2.3. The Issue of Optimality

Theorem 1 settles for an embedding of complete binary trees in Butterfly graphs, that achieves dilation $O(1)$ and expansion $O(1)$ simultaneously. While this achieves our overall goal of optimality to within constant factors, it does leave open the possibility of those constant-factor improvements. We have been unable to determine exact dilation-expansion tradeoffs for embeddings of complete binary trees in Butterfly graphs, but we can show easily that it is impossible to optimize both cost

measures simultaneously. Thus, one cannot hope for the level of "perfection" found in, say, [GHR][6].

Proposition 2 *No embedding of $T(q)$ in $B(m+1)$ has unit dilation.*

Proof. Both complete binary trees and Butterfly graphs are bipartite graphs: one can color the vertices of either graph red and blue in such a way that every edge connects a red vertex and a blue one. For any Butterfly graph $B(r)$, on the one hand, the numbers of red and blue vertices are within r of being equal; for any complete binary tree, on the other hand, one of the sets has roughly twice as many vertices as the other. Thus, one cannot find a unit-dilation embedding of a complete binary tree in the smallest Butterfly graph that has enough vertices to hold it. □

3. UPPER BOUNDS – THEOREM 2

This section is devoted to proving Theorem 2. Since all of the relevant ideas in the proof are present in its application to specific families of graphs, we actually prove only the upper bound of Corollary 1. The reader should be able to generalize easily to arbitrary families of graphs, thereby proving Theorem 2. For the remainder of the Section, we therefore focus on the problem of embedding X-trees in Butterflies.

Our embedding of the X-tree in the Butterfly graph is indirect: First we find a unit-expansion, dilation-$O(\log\log n)$ embedding of $X(h)$ in $T(h)$. Then we compose this embedding with the expansion-$O(1)$, dilation-$O(1)$ embedding of $T(h)$ in $B(m)$ from Theorem 1, to obtain the upper bound of Theorem 2. We discuss here only the former embedding, which, in fact, embeds the X-tree $X(m)$ in the complete binary tree $T(m)$. For notational simplicity, let $n =_{\text{def}} 2^{m+1} - 1$, the number of vertices in $X(m)$. We devote this section to proving the following.

Proposition 3 *For any integer m, one can embed the X-tree $X(m)$ in the complete binary tree $T(m)$, with dilation $O(\log m) = O(\log\log n)$.*

Using the obvious fact that the n-vertex X-tree can be bisected (in the sense of statement 1 above) by removing $O(\log n)$ edges, coupled with techniques in Section 1 of [BL], the reader can easily prove the following.

[6] In [GHR] a variant of $B(m)$ with no wraparound is embedded in the Hypercube with unit dilation and optimal expansion.

Lemma 1 *For all positive integers n, k, the n-vertex X-tree has a k-color $\sqrt{2}$-bifurcator of size $S = 2k \cdot \log n$.*

Proof of Proposition 3. Our embedding uses the following auxiliary structure, which appears (in slightly different form) in [BCLR]. A *bucket tree* is a complete binary tree, each of whose level-ℓ vertices has *(bucket) capacity*

$$c \cdot \log\left(\frac{n}{2^\ell}\right)$$

for some fixed constant c to be chosen later (in Lemma 2). We embed $X(m)$ in $T(m)$ in two stages: First, we embed $X(m)$ in a bucket tree, via a many-to-one function μ that "respects" bucket capacities (always placing precisely $c \cdot \log((2^{m+1} - 1)/2^\ell)$ vertices of $X(m)$ in each level-ℓ vertex of the bucket tree) and has constant "dilation". Then we "spread" the contents of the bucket tree's buckets within $T(m)$, to achieve an embedding of $X(m)$ in $T(m)$, with the claimed dilation. Formally, the first stage of the embedding is described as follows.

Lemma 2 *Every X-tree $X(m)$ can be mapped onto a bucket tree in such a way that:* (a) *exactly*

$$N(\ell) = 14 \log\left(\frac{2^{m+1} - 1}{2^\ell}\right) + 24$$

vertices of $X(m)$ are mapped to each level-ℓ vertex of the bucket tree, and (b) *vertices that are adjacent in $X(m)$ are mapped to buckets that are at most distance 5 apart in the bucket tree.*

> The constants in the expression for $N(\ell)$ can be reduced by increasing the constant 5 in part (b) of the Lemma (say, to 10). We suffer the larger constants in order to simplify the technical development in the proof. The interested reader can easily mimic our development with other constants.

Proof. The basic idea is to recursively bisect $X(m)$, using a 5-color $\sqrt{2}$-bifurcator (the uses of the colors will become clear momentarily), placing successively smaller sets of $\sqrt{2}$-bifurcator vertices in lower-level buckets of the bucket tree. We also place other vertices in the buckets, in order to ensure the desired "dilation" and in order to ensure that all buckets are filled to capacity. The formal description of the mapping will require two iterations. First, we present a mapping procedure that establishes the sufficiency of the quantities $N(\ell)$ as bucket capacities. Then we refine the initial mapping to complete the proof.

We simplify our description of this technically cumbersome procedure in two ways. First, we describe in detail what the procedure would look like if we were using *9-color bifurcators* rather than 5-color bifurcators; the reader should be able to extrapolate from oor description to arbitrray numbers of colors. Second, we establish the following notation.

- We denote by B_λ, where λ denotes the null string (i.e., the string of length 0) over the alphabet $\{1,2\}$, the bucket at the root of the bucket tree.
- In general, letting x denote any string over the alphabet $\{1,2\}$, we denote by B_{x1} and B_{x2} the buckets at the children of the vertex of the bucket tree having bucket B_x; for example, B_1 and B_2 denote the buckets at the children of the root vertex of the bucket tree, B_{11} and B_{12} denote the buckets at the left grandchildren of the root vertex, B_{21} and B_{22} denote the buckets at the right grandchildren of the root vertex, and so on.

Algorithm Bucket: Mapping $X(m)$ into a bucket tree

Step 1. Initial coloring and bisection.

1.a. Initialize every vertex of $X(m)$ to color A.

1.b. Associate[7] the graph $X(m)$ with the root of the bucket tree.

1.c. Bisect $X(m)$, to obtain subgraphs X_1 and X_2, and place the $\sqrt{2}$-bifurcator vertices in bucket B_λ.

1.d. Recolor every A-colored vertex of $X(m)$ that is adjacent to a vertex in bucket B_λ with color 0.

1.e. Associate X_i ($i \in \{1,2\}$) with the child of the root vertex of the bucket tree holding bucket B_i.

Step 2. Second-level bisection.

2.a. Use a 2-color $\sqrt{2}$-bifurcator for each X_i, to create subgraphs X_{i1} and X_{i2}.

2.b. Place the $\sqrt{2}$-bifurcator vertices for each X_i in the corresponding bucket B_i of the bucket tree.

2.c. Recolor every A-colored vertex of $X(m)$ that is adjacent to a vertex in bucket B_i with color 1.

[7] The "associations" here are intended to make it easier for the reader to follow our description of the mapping.

2.d. For each X_i, associate each subgraph X_{ij} with the $\sqrt{2}$-bifurcator-tree vertex associated with bucket B_{ij}.

Step 3. Third-level bisection.

3.a. Use a 3-color $\sqrt{2}$-bifurcator for each X_{ij}, to create subgraphs X_{ij1} and X_{ij2}.

3.b. Place the $\sqrt{2}$-bifurcator vertices for each X_{ij} in the corresponding bucket B_{ij} of the bucket tree.

3.c. Recolor every A-colored vertex of $X(m)$ that is adjacent to a vertex in bucket B_{ij} with color 0.

3.d. For each X_{ij}, associate each subgraph X_{ijk} with the $\sqrt{2}$-bifurcator-tree vertex associated with bucket B_{ijk}.

Step s. ($4 \leq s \leq m$) All remaining bisections.

s.a. For each subgraph X_y ($y \in \{1,2\}^*$) of $X(m)$ created in Step $s-1$, place every vertex of color $s \pmod 2$ in the associated bucket B_y.

s.b. Use a 3-color $\sqrt{2}$-bifurcator for each X_y, to create subgraphs X_{y1} and X_{y2}.

s.c. Place the $\sqrt{2}$-bifurcator vertices for each X_y in the corresponding bucket B_y of the bucket tree.

s.d. Recolor every A-colored vertex of $X(m)$ that is adjacent to a vertex in bucket B_y with color $length(y) \pmod 2$.

s.e. For each X_y, associate each subgraph $X_{y\ell}$ with the $\sqrt{2}$-bifurcator-tree vertex associated with bucket $B_{y\ell}$.

We now analyze *5-color analogue of the described mapping*, to show that it satisfies the demands of Lemma 2, with the requirement of "exactly" $N(\ell)$ vertices per level-ℓ bucket replaced by "no more than" $N(\ell)$ vertices per level-ℓ bucket, i.e., to show that our bucket capacities are big enough. Since the "dilation" condition (b) is transparently enforced when certain colored vertices are automatically placed in buckets (in Step s.a), it will suffice to establish that the populations of the buckets are as indicated in the modified condition (a). This follows by the following recurrence, wherein $N(k)$ denotes the number of vertices of $X(m)$ that get mapped into a bucket at level $k-1$ of the bucket tree.

$$\begin{aligned} N(k) &\leq \left\lceil \frac{5}{32} N(k-5) \right\rceil + 6 \log \left(\frac{n}{2^k} \right) \\ &\leq \frac{3}{16} N(k-5) + 10 \log \left(\frac{n}{2^k} \right) \end{aligned}$$

with initial conditions

- $N(1) \leq 2 \log n$
- $N(2) \leq 4 \log \left(\frac{n}{2}\right)$
- $N(3) \leq 6 \log \left(\frac{n}{4}\right)$
- $N(4) \leq 8 \log \left(\frac{n}{8}\right)$
- $N(5) \leq 10 \log \left(\frac{n}{16}\right)$

The initial conditions reflect the sizes of the appropriately colored $\sqrt{2}$-bifurcators of $X(m)$: At each level ℓ, $1 \leq \ell \leq 4$, one uses an ℓ-colored $\sqrt{2}$-bifurcator, followed by a 5-color $\sqrt{2}$-bifurcator at all subsequent levels. At levels $s > 2$, the buckets contain not only $\sqrt{2}$-bifurcator vertices, which account for the term

$$10 \log \left(\frac{n}{2^k}\right)$$

in the general recurrence; they contain also the vertices of $X(m)$ that are placed to satisfy the "dilation" requirements. The latter vertices comprise all neighbors of the $N(k-5)$ occupants of the distance-4 ancestor bucket that have not yet been placed in any other bucket. Since vertices of $X(m)$ can have no more than five neighbors, and since our 5-color bisections allocate these neighbors equally among the descendants of a given bucket, these "dilation"-generated vertices can be no more than

$$\left\lceil \frac{5}{32} N(k-5) \right\rceil \leq \frac{3}{16} N(k-5)$$

in number. These two sources, the $\sqrt{2}$-bifurcators and their neighbors, account for the occupants of the buckets and for the recurrence counting them. To complete the proof of the modified Lemma, one now shows by standard techniques that the indicated recurrence, with the indicated initial conditions, has the solution

$$N(k) \leq 14 \log \left(\frac{n}{2^k}\right) + 24.$$

Finally, we turn to the original form of the Lemma. This follows from the modified form, upon refining the Algorithm by adding the following substeps at the indicated points.

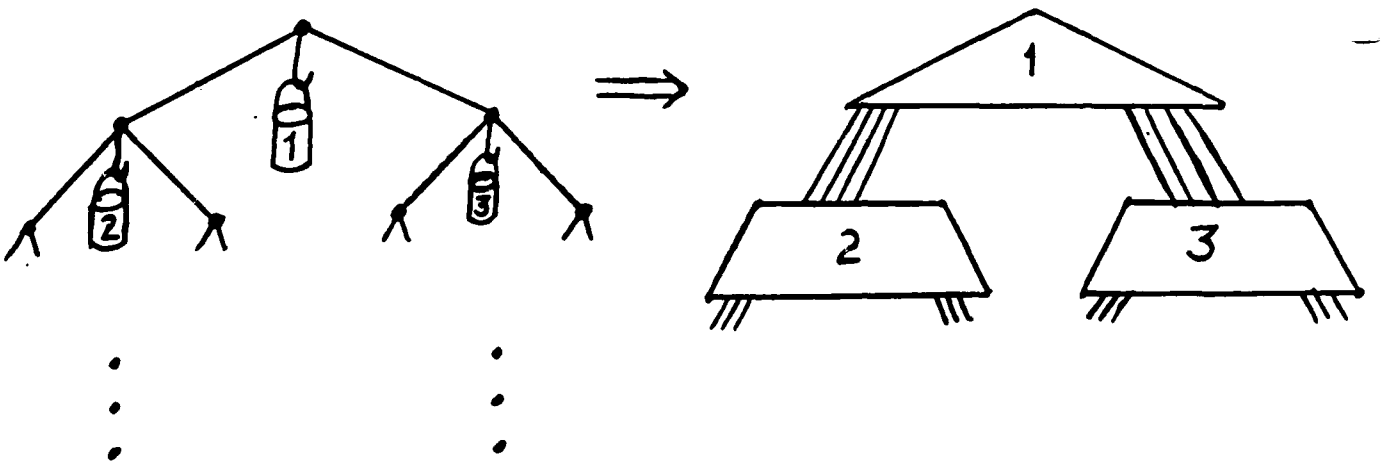


Figure 7: Unloading the buckets

> At the end of each step of the Algorithm, when we have finished filling a bucket B_x ($x \in \{1,2\}^*$) with vertices obtained from a recent bisection or from our desire to maintain small "dilation", we check the population of the bucket against the ceiling population $N(\ell)$, where $\ell = \mathit{length}(x)$. If the bucket contains fewer than $N(\ell)$, vertices, then we add enough new vertices to it from the remaining associated subgraph to fill it to capacity.

This last step ensures that all buckets at level ℓ of the bucket tree contain exactly $N(\ell)$ vertices. □

Our final task is to refine the "dilation"-5 mapping of Lemma 2 to a bona fide embedding of $X(m)$ in $T(m)$, having dilation $O(\log\log n)$. We proceed inductively, emptying buckets into $T(m)$ in such a way that each tree vertex is assigned a unique X-tree vertex. In general, we denote by T_x the smallest subtree of $T(m)$ that is rooted at level $\mathit{length}(x)$ of $T(m)$ and that contains the contents of bucket B_x. (In general, the contents of B_x will occupy only the last few levels of T_x.) See Fig. 7.

- Place the $\log n$ elements of bucket B_λ in the topmost copy of $T(\log\log n)$ in $T(m)$, in any way.
- Consider the subtrees of T_λ rooted at level 1 of $T(m)$. Place the contents of bucket B_1 in the (roughly) $\log\log n$ levels of the leftmore of these two subtrees,

starting immediately after the leaves of T_λ. Place the contents of bucket B_2 analogously, using the rightmore of these two subtrees, starting immediately after the leaves of T_λ. We have thus implicitly defined the subtrees T_1 and T_2.

Note that by this point, we are using enough of the top levels of $T(m)$ that we need use only one more level in order to place the contents of the next level of buckets. The importance of this fact is that it guarantees that all of the subtrees T_x will have height $O(\log\log n)$. (Namely, T_λ, T_1, and T_2 have the desired height, and all subsequent trees will result from adding one level of leaves to a tree whose root is one level lower in $T(m)$ than was its father's root.)

- Proceeding inductively, assume that we have filled subtrees T_x of $T(m)$ with bucket contents, for strings $x \in \{1,2\}^*$ of length $\leq \ell$. We now consider the subtrees of $T(m)$ rooted at level $\ell+1$; each subtree T_x rooted at level ℓ thus spawns two children. We order these $2^{\ell+1}$ subtrees from left to right, according to the lexicographic order on the subscript-strings x. We then place the contents of the bucket B_{x1} in the leaves of the leftmore of the children of T_x, beginning where the contents of bucket B_x left off. Analogously, we place the contents of the bucket B_{x2} in the leaves of the rightmore of the children of T_x, beginning where the contents of bucket B_x left off.

The described procedure clearly produces an embedding of $X(m)$ in $T(m)$, since each vertex of $X(m)$ is assigned to a unique tree vertex. Additionally, the embedding has unit expansion since no tree vertices are passed over in the assignment process and since all buckets at each level ℓ have the same population $N(\ell)$ (so all subtrees T_x are isomorphic). Finally, the procedure's method of spreading bucket contents throughout $T(m)$ produces an embedding with the desired dilation, namely, $O(\log\log n)$. Specifically, by always spreading the contents of buckets B_{x1} and B_{x2} in the leaves of the left and right subtrees of the depth-$O(\log\log n)$ subtree that contains the contents of bucket B_x, the procedure guarantees that the least common ancestor, in $T(m)$, of the set comprising the contents of any bucket plus the vertices in buckets at most five buckets up (which will lie in adjacent levels $k, k+1, k+2, k+3, k+4, k+5$ of the bucket tree) are always within a subtree of height $O(\log\log n)$ of $T(m)$. Thus, we have produced the desired embedding, thereby proving Proposition 2, hence Theorem 2. □

4. LOWER BOUNDS – THEOREM 3

We demonstrate the near-optimality (to within constant factors) of the embeddings of Section 3 - in fact, true optimality for X-trees - by proving the lower bound of

Theorem 3. In contrast with Theorem 2, Theorem 3 is most easily proved in its full generality.

Assume henceforth that we are given a planar graph G, a planar embedding ϵ of G, and a minimum-dilation embedding μ of G in $B(p)$; let μ have dilation δ.

We begin by noting that we can simplify our quest somewhat. Specifically, since we aim only for bounds that hold up to constant factors, we lose no generality by assuming henceforth that (in the embedding ϵ) the exterior face of G is a simple cycle:

Lemma 3 *One can add edges to the graph G within the embedding ϵ in such a way that*

- *the resulting embedding ϵ' is a planar embedding of the resulting graph G'*
- *in the embedding ϵ', the exterior face of G' is a simple cycle*
- $\Sigma(G') = O(\Sigma(G))$
- $\Phi_{\epsilon'}(G') = \max(3, \Phi_{\epsilon}(G))$.

Proof Sketch. If the exterior face of G is not a simple cycle, it is because of cut-edges and/or pinch-vertices. We take each cut-edge in turn and create a triangle containing it as an edge; then we repeat the process with any remaining cut-edge. When no more cut-edges exist, we eliminate each pinch-vertex in turn by creating a triangle that includes the pinch-vertex as a vertex. Since each added edge creates a triangle and spans only two edges of G, the claims about $\Phi(G')$ and $\Sigma(G')$ are immediate. □

A consequence of Lemma 3 is that we may henceforth assume that every edge of G resides in some interior face (in the embedding ϵ).

We turn now to the quantitative consequences of Lemma 3.

A set of faces of G is *connected* in the embedding ϵ just when their corresponding vertices are connected in the graph $\Gamma(G;\epsilon)$ whose vertices are the faces of G and whose edges connect a pair of face-vertices just when the faces share a vertex. A set S of vertices of G is *face-connected* (in ϵ) if the set of *interior faces* of G that contain one or more of the vertices of S is connected.

Let A be a connected component of the graph G remaining after removing a set S of vertices from G. The *S-boundary of A* is the set of vertices of A that are adjacent (in G) to vertices of S.

Lemma 4 *If one removes a face-connected set of vertices S from the graph G, then the S-boundary of every resulting maximal connected component of G is face-connected.*

Proof. Consider a maximal connected component A remaining after removing S from G. Assume for contradiction that the set of S-boundary vertices of A is not face-connected. There must then be at least two distinct maximal connected components, call them F_1 and F_2, of interior faces that contain boundary vertices (so $F_1 \cup F_2$ is not connected). Let f_i, $i = 1, 2$, be an interior face in component F_i, and let b_i be a boundary vertex in face f_i. Since each edge of G lies in an interior face, we can choose each f_i to contain a vertex of S as well as a boundary vertex.

Fact 1 *There is a connected set I of interior faces, none of which contains a boundary vertex, such that I separates f_1 from f_2.*

Verification. It is not possible for both F_1 to encircle f_2 and F_2 to encircle f_1, since then F_1 and F_2 would intersect (so f_1 and f_2 would be connected by interior faces containing boundary vertices). Without loss of generality, say that F_1 does not encircle f_2.

Let J be the set of interior faces that do not contain boundary vertices and that are incident to the outer boundary of F_1 (so that f_2 is on the outside). By definition, the set J separates f_1 from f_2. If J is connected, then it is the desired set I. If J is not connected, then adding the exterior face of G to J yields a connected set J'. Moreover, f_2 must lie in one of the simply connected regions J'' of J'. Deleting the exterior face from J'' then yields the desired set I; see Fig. 8.

Fact 2 *I contains a vertex of A and a vertex of S.*

Verification. I separates f_1 from f_2, yet: f_1 and f_2 both contain vertices of both A and S; both A and S are face-connected in G.

Since I contains vertices of A and S and is connected, and since S separates the connected set A from the rest of G, the set I must contain at least one face that contains *both* a vertex from A and a vertex from S. Such a face must also contain a vertex of the S-boundary of A, contradicting Fact 1. Lemma 4 follows. □

A set of vertices S of a graph K is *d-quasi-connected*, d a positive integer, if for every two vertices u, w of S, there exists a chain of vertices

$$u = v_0, v_1, v_2, \ldots, v_k = w,$$

of S, where consecutive vertices v_i, v_{i+1} are distance $\leq d$ apart in K.

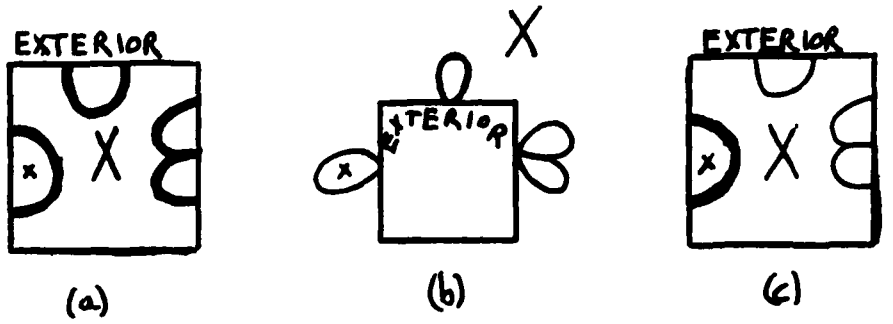


Figure 8: (*a*) The embedding ϵ, with the set J outlined boldly; "x" marks f_2, and "X" marks F_2 with the holes filled in. (*b*) The set $J' = J \cup$ (outer face). (*c*) The embedding ϵ, as in (a), with the set J'' outlined boldly.

Lemma 5 The Fundamental Lemma for Butterfly-Like Graphs
Say that there is a subgraph K of G and a constant c such that

- *K is $\Phi(G)$-quasi-connected*
- *the image of K under the embedding μ lies within $c\Phi(G)\delta$ consecutive levels of $B(p)$.*

Then $\delta \geq \alpha(c)\frac{\log|K|}{\Phi(G)}$, where $\alpha(c)$ is a constant depending only on c.

Proof. Say that the image of H under μ lies entirely in levels[8]

$$l+1, l+2, \cdots, l+c\Phi(G)\delta$$

of $B(p)$. Let u and v be arbitrary vertices of H which are connected by a path of at most $\Phi(G)$ vertices in G. The image of this path in $B(p)$ must lie totally within levels

$$l-\Phi(G)\delta+1, \cdots, l+(c+1)\Phi(G)\delta$$

of $B(p)$, since the embedding μ has dilation δ. See Fig. 9. Since K is $\Phi(G)$-quasi-connected, this means that the PWL strings of *all* images of vertices of K can differ

[8] All addition is modulo p.

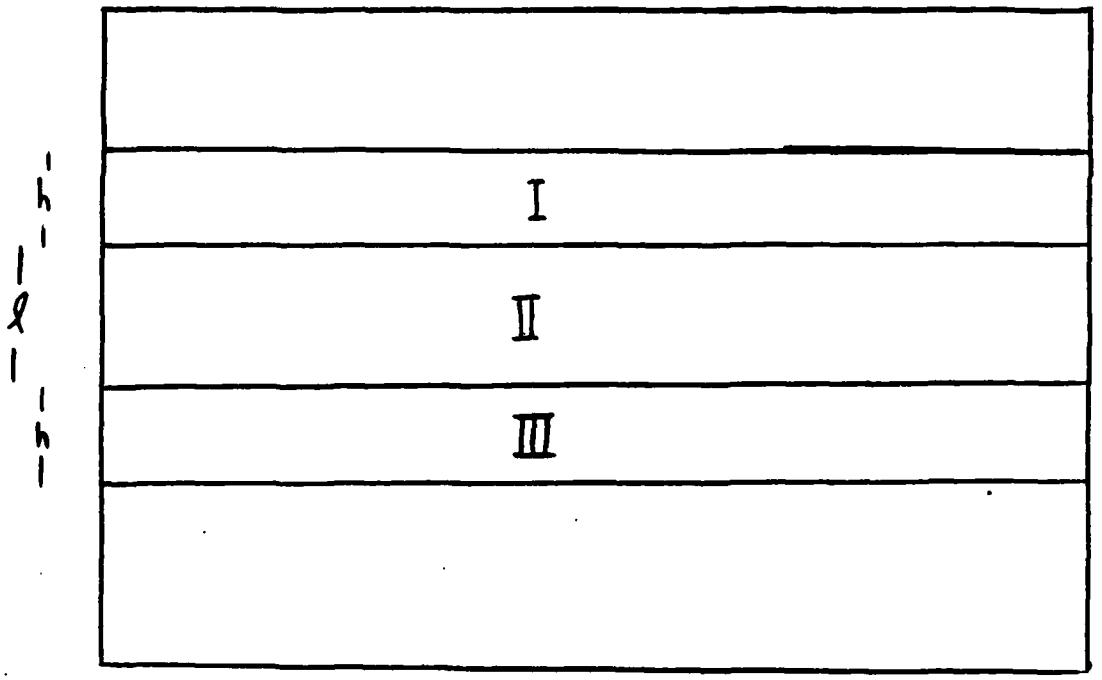


Figure 9: Illustrating the Fundamental Lemma, with $\ell = c\Phi(G)\delta$ and $h = \Phi(G)\delta$: Vertices of K reside in region II; length-$\Phi(G)$ paths between vertices of K cannot extend beyond regions I or III.

only in some set of at most $((c+2)\Phi(G)+1)\delta$ bit positions. It follows that K can contain no more than $c\Phi(G)\delta 2^{((c+2)\Phi(G)+1)\delta}$ vertices, i.e., $c\Phi(G)\delta$ levels of $B(p)$, with at most $2^{((c+2)\Phi(G)+1)\delta}$ vertices per level. In other words,

$$c\Phi(G)\delta 2^{((c+2)\Phi(G)+1)\delta} \geq |K|,$$

whence the result. □

We now complete the proof of Theorem 3, beginning with two simple lemmas.

Lemma 6 *Any face-connected set of vertices of G is $\Phi(G)$-quasi-connected.*

Proof Sketch. Any vertex in an f-vertex face is distance $\leq f/2$ from any neighboring face. □

Lemma 7 *Let C be a set of vertices of the graph G whose removal partitions G into connected components all of size $\leq |G|/2$. Then C is a 1/3-2/3 separator of G.*

Proof Sketch. Remove C from G, order the resulting connected components by size into decreasing order, and lump the components into two piles as follows.

- Place the largest component into the left pile.
- Place as few of the largest remaining components in the right pile as possible until the right pile is bigger than the left.
- Now alternate piles, adding as few of the largest remaining piles as possible to the smaller pile until the smaller first becomes bigger than the larger pile.

Clearly, when one has completed the two piles, the larger cannot be bigger than the smaller by more than the size of the third largest component, i.e., by more than $|G|/3$ vertices. It follows that each pile must contain at least $|G|/3$ vertices, whence the claim. □

Theorem 3 will now follow from the next Lemma.

Lemma 8 *The embedding μ must have dilation $\delta \geq (const)\left(\frac{\log \Sigma(G)}{\Phi(G)}\right)$.*

Proof. Partition $B(p)$ into *bands*, each band β_i being a sequence of $d_i\delta$ consecutive levels, $2\Phi(G) \leq d_i < 4\Phi(G)$, where the constants d_i may be chosen in any way that achieves a partition. Let $\kappa(v)$, the *color* of vertex v of G, be the index i of the band β_i in which $\mu(v)$ resides.

We perform a modified breadth-first search of G, to find a $\Phi(G)$-quasi-connected component of size $\geq \Sigma(G)$, all of whose vertices have images in a single band of $B(p)$, hence the same color. By Lemma 5, the existence of such a component will yield the lower bound on δ.

The breadth-first search proceeds as follows. We select an arbitrary vertex v_0 of G and form V_0, the maximal connected component of G that contains v_0 and that consists entirely of vertices with color $\kappa(v_0)$. Since V_0 is connected, removing its vertices partitions G into connected components; let C_0 be the largest of these. Lemmas 4 and 6 assure us that the V_0-boundary, B_0, of the component C_0 is $\Phi(G)$-quasi-connected. It follows that

Fact 3 *All vertices of B_0 have the same color.*

Verification. Since each $v \in B_0$ is adjacent to a vertex of V_0, we must have $\kappa(v) \in \{\kappa(v_0) - 1, \kappa(v_0) + 1\}$. Moreover, B_0 cannot contain vertices of both colors: Two such vertices would be separated by the band $\beta_{\kappa(v_0)}$, contradicting the fact that B_0 is $\Phi(G)$-quasi-connected.

Next, form V_1, the maximal *monochromatic* subgraph of G that contains both B_0 and all connected components of G that intersect B_0; obviously, V_1 is $\Phi(G)$-quasi-connected, so removing it partitions G into some number of connected components. Let C_1 be the largest of these, and let B_1 be the V_1-boundary of C_1. As with B_0, one shows that B_1 is $\Phi(G)$-quasi-connected and monochromatic.

We continue in this fashion, constructing, in turn, for $i = 2, 3, \ldots$, the following subgraphs of G, with the indicated properties:

- V_i: the ($\Phi(G)$-quasi-connected) maximal monochromatic subgraph of G that contains both B_{i-1} and all connected components of G that intersect B_{i-1}
- C_i: the largest connected component of G remaining when one removes V_i from G
- B_i: the ($\Phi(G)$-quasi-connected, monochromatic) V_i-boundary of C_i

One continues this construction until some subgraph V_i contains at least $\Sigma(G)$ vertices. We now show that this point must occur.

Fact 4 *For some i, $|V_i| \geq \Sigma(G)$.*

Verification. Note that at each point in our construction, V_i is whittled out of the largest component C_{i-1} of G remaining after removal of V_{i-1} from G. Moreover, V_i disconnects the vertices of $C_{i-1} - V_i$ from the remainder of G, as one can verify easily by induction on i. At some point, therefore, the whittling process must reduce the size of the then-current largest component C_m so that $|C_m| \leq |G|/2$. By Lemma 7, the then-current V_n is a 1/3-2/3 separator of G, hence must contain at least $\Sigma(G)$ vertices.

The preceding development gives us a set of vertices, of size $\geq \Sigma(G)$, whose images reside in a single band of d_i levels of $B(p)$. By Lemma 5, Theorem 3 follows. □

5. THE COROLLARIES: X-TREES AND MESHES

Corollaries 1 and 2 now follow from the following Lemmas.

Lemma 9 $|\mathrm{HR}|\ \Sigma(X(h)) = \Omega(h) = \Omega(\log|X(h)|)$, *and* $\Phi(X(h)) = 5$ *(under the natural embedding).*

Lemma 10 (e.g., $|\mathrm{HR}|$) $\Sigma(M(s)) = \Omega(s) = \Omega(\sqrt{|M(s)|})$, *and* $\Phi(M(s)) = 4$ *(under the natural embedding).*

6. CONCLUDING REMARKS

We close with some remarks about extensions to the research described here.

The lower bound of Theorem 3 cannot be improved in general, as one can see from considering homeomorphs of the mesh.

Our lower bound for the mesh extends also to higher-dimensional meshes and to pyramid graphs; thus, these are examples of other popular networks that embed efficiently in the Hypercube, but not in butterfly-like machines.

The lower bound of Theorem 3, which deals explicitly only with embeddings in the Butterfly, extends to embeddings in the mesh of trees, Cube-Connected-Cycles, Benes network, and similar levelled networks.

We do not yet have an analogue of Theorem 3 for embeddings in the shuffle-exchange and deBruin graphs[9]. However, using rather complicated arguments, we can prove that any expansion-$O(1)$ embedding of the n-vertex X-tree or the n-vertex mesh in these host graphs requires dilation $\Omega(\log \log n)$. Since a complete binary tree is a spanning tree of the deBruijn graph, the proof technique of Section 3 shows that this lower bound for the X-tree is optimal. We suspect that the lower bound for the mesh can be improved.

In order to justify dilation fully as the central measure of concern in network embeddings, it would be nice to strengthen the results of Section 3 to show that the Butterfly can simulate any graph having a $\sqrt{2}$-bifurcator of size $S = \Omega(\log n)$ with delay $O(\log S)$. We believe this to be possible using the arguments of Section 2, but we have not worked through the details.

Lastly, it should be noted that our lower bounds do *not* mean that a Butterfly cannot efficiently simulate a mesh or X-tree efficiently over a large span of time. For example, a Butterfly can simulate $\log n$ steps of a mesh of a constant fraction smaller size within $O(\log n \log \log n)$ steps, and possibly within $O(\log n)$ steps. Similar improvements in *amortized* simulation times are also possible for the X-tree, and we are currently studying how good such amortized simulations can be in general.

ACKNOWLEDGMENTS: The authors wish to thank Dave Barrington and Les Valiant for helpful conversations.

7. REFERENCES

[Ag | A. Aggarwal (1984): A comparative study of X-tree, pyramid, and related machines. *25th IEEE Symp. on Foundations of Computer Science*, 89-99.

[ABR | F. Annexstein, M. Baumslag, A.L. Rosenberg (1987): Group-action graphs and parallel architectures. Tech. Rpt., Univ. of Massachusetts; submitted for publication.

[Be | V.E. Benes (1964): Optimal rearrangeable multistage connecting networks. *Bell Syst. Tech. J. 43*, 1641-1656.

[BCLR | S.N. Bhatt, F.R.K. Chung, F.T. Leighton, A.L. Rosenberg (1986): Optimal simulations of tree machines. *27th IEEE Symp. on Foundations of Computer Science*, 274-282.

[9] These are "essentially" the same graph, since each can be embedded in the other with unit expansion and dilation 2.

[BI | S.N. Bhatt and I. Ipsen (1985): Embedding trees in the hypercube. Yale Univ. Rpt. RR-443.

[BL | S.N. Bhatt and F.T. Leighton (1984): A framework for solving VLSI graph layout problems. *J. Comp. Syst. Sci. 28*, 300-343.

[DP | A.M. Despain and D.A. Patterson (1978): X-tree – a tree structured multiprocessor architecture. *5th Symp. on Computer Architecture*, 144-151.

[Ga | D. Gannon (1980): On pipelining a mesh-connected multiprocessor for finite element problems by nested dissection. *Intl. Conf. on Parallel Processing.*

[GHR | D.S. Greenberg, L.S. Heath, A.L. Rosenberg (1987): Optimal embeddings of the FFT graph in the Hypercube. Typescript, Univ. of Massachusetts; submitted for publication.

[HR | J.-W. Hong and A.L. Rosenberg (1982): Graphs that are almost binary trees. *SIAM J. Comput. 11*, 227-242.

[HZ | E. Horowitz and A. Zorat (1981): The binary tree as an interconnection network: applications to multiprocessor systems and VLSI. *IEEE Trans. Comp., C-30*, 247-253.

[Le | F.T. Leighton (1984): Parallel computation using meshes of trees. *1983 Workshop on Graph-Theoretic Concepts in Computer Science*, Trauner Verlag, Linz, pp. 200-218.

[PV | F.P. Preparata and J.E. Vuillemin (1981): The cube-connected cycles: a versatile network for parallel computation. *C. ACM 24*, 300-309.

[Ro | A.L. Rosenberg (1981): Issues in the study of graph embeddings. In *Graph-Theoretic Concepts in Computer Science: Proceedings of the International Workshop WG80*, Bad Honnef, Germany (H. Noltemeier, ed.) *Lecture Notes in Computer Science 100*, Springer-Verlag, NY, 150-176.